**Design Summary:**

Tokens are described as following:

**Keywords**

|  |  |
| --- | --- |
| Integer, character, float | Data Type |
| ForLoop | For loop Statement |
| Switch, case default | Switch case default statement |
| if | Conditional “if” Statement |
| else | Conditional “else” Statement |
| break | Loop end Statement |
| main\_function | Main function |

**Operators**

|  |  |  |
| --- | --- | --- |
| **Operators** | **Operation** | **Example** |
| + | addition | a + b |
| - | subtraction | a - b |
| \* | multiplication | a \* b |
| / | division | a / b |
| > | Greater than | a > b |
| < | Less than | a < b |
| = | Assign | a = b |
| ^ | power | a ^ b |
| sin | sine | sin ( 30 ) |
| cos | cosine | cos ( 30 ) |
| tan | tangent | tan ( 30 ) |
| log | Logarithm | log ( 30 ) |

**Grammar**

|  |  |
| --- | --- |
| program: | MAIN '(' ')' START cstatement END |
| cstatement: | /\* NULL \*/  | cstatement statement  ; |
| statement: | ';'|declaration';'|expression ';'|VAR '=' expression ';'{ sym[$1] = $3; }  | SWITCH '(' VAR ')' START Bases END  | FOR '(' NUM THREEDOT NUM ')' START statement END  | IF '(' expression ')' START statement END %prec IFX  | IF '(' expression ')' START statement END ELSE START statement END; |
| expression: | NUM { $$ = $1; }  | VAR { $$ = sym[$1]; }  | expression '+' expression { $$ = $1 + $3; }  | expression '-' expression { $$ = $1 - $3; }  | expression '\*' expression { $$ = $1 \* $3; }  | expression '/' expression { $$ = $1 / $3; }  | '-' expression %prec UMINUS { $2 = - $2; }  | expression '^' expression { $$ = pow($1, $3); }  | expression '<' expression { $$ = $1 < $3; }  | expression '>' expression { $$ = $1 > $3; }  | '(' expression ')' { $$ = $2; }  | SIN '(' expression ')' { double val = sin($3 \* PI / 180); printf("sin %d = %lf\n",$3, val); $$ = val;}  | COS '(' expression ')' { double val = cos($3 \* PI / 180); printf("cos %d = %lf\n",$3, val); $$ = val; }  | TAN '(' expression ')' { double val = tan($3 \* PI / 180); printf("tan %d = %lf\n",$3, val); $$ = val; }  | LN '(' expression ')' { double val = log($3 \* PI / 180); printf("ln %lf\n",$3, val); $$ = val; }  ; |
| Bases: | Base  | Base Defaults ; |
| Base: | /\*NULL\*/  | Base Cases; |
| Cases: | CASE NUM ':' expression ';' BREAK ';' { printf("Value of the case expression: %d\t\n",$4); } ; |
| Defaults: | DEFAULT ':' expression ';' BREAK ';' { printf("Value of the default expression: %d\t\n",$3); }; |
| Declaration: | TYPE ID1 ; |
| TYPE: | INT  | FLOAT  | CHAR  ; |
| ID1: | ID1 ',' VAR  |VAR  ; |

Source Program

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/\*

This is a simple source file.

Author: MD. Ahsanul Haque

\*/

main\_function()

start

//Starting main function of source program

//Variable declaration source program

integer a, b, c, i, j;

float f, g, h;

//Assignment statement source program

a = 10;

b = 20;

c = 30;

// expresssion evaluation

sin(60);

cos(60);

tan(60);

tan(60);

2+2\*2^3;

2+-5;

i = a + b;

j = 10;

//if else source program

if(i < 3)

start

if(j < 1)

start

a + 2;

end

else

start

c + 3;

end

end

else

start

b + 6;

end

// Switch case source program

switch ( a )

start

case 1 :

a + b;

break;

case 2 :

b + c;

break;

case 3 :

a + c;

break;

default :

a + a;

break;

end

// Printing the values of the variables

a;

b;

c;

i;

a = 5;

//For loop source program

ForLoop(1...5)

start

a + 1;

end

end

Flex.l file

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/\* C Declarations \*/

%{

#include<stdio.h>

#include "test.tab.h"

#include<stdlib.h>

extern int yylval;

%}

/\* RE and Actions \*/

SINGLELINECOMMENT "//".\*

MULTLINECOMMENT [/][\*][^\*]\*[\*]+([^\*/][^\*]\*[\*]+)\*[/]

DIGIT [0-9]

NUMBER [-]?{DIGIT}\*[.]?{DIGIT}+

%%

[a-z] {

yylval = \*yytext - 'a';

return VAR;

}

{NUMBER} {

yylval = atoi(yytext);

return NUM;

}

{SINGLELINECOMMENT} {printf("\nSingle Line Comment\n"); }

{MULTLINECOMMENT} { printf("\nMulti Line Comment\n");}

"ForLoop" { return FOR; }

"..." { return THREEDOT; }

"switch" { return SWITCH; }

"case" { return CASE; }

"break" { return BREAK; }

"default" { return DEFAULT;}

"start" { return(START); }

"end" { return(END); }

"integer" { return(INT); }

"character" { return(CHAR); }

"float" { return(FLOAT); }

"if" { return IF; }

"else" { return ELSE; }

"main\_function" { return MAIN; }

"sin" { return SIN; }

"cos" { return COS; }

"tan" { return TAN; }

"ln" { return LN; }

[-+/\*^<>=,{}():;] {

yylval = yytext[0];

return \*yytext;

}

[ \t\n]\* ;

. {

yyerror("Unknown Character.\n");

}

%%

main(){

yyin = freopen("in1.txt","r",stdin);

yyout = freopen("out.txt","w",stdout);

yyparse();

}

Bison file

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/\* C Declarations \*/

%{

#include<stdio.h>

#include<math.h>

#define PI 3.141593

int sym[26];

int temp;

%}

/\* bison declarations \*/

%token NUM VAR IF ELSE MAIN INT FLOAT CHAR START END SWITCH CASE DEFAULT BREAK

%token SIN COS TAN LN THREEDOT FOR

%nonassoc IFX

%nonassoc ELSE

%nonassoc SWITCH

%nonassoc CASE

%nonassoc DEFAULT

%left UMINUS

%left '<' '>'

%left '+' '-'

%left '\*' '/'

%right '^'

/\* Grammar rules and actions follow. \*/

%%

program: MAIN '(' ')' START cstatement END

;

cstatement: /\* NULL \*/

| cstatement statement

;

statement: ';'

| declaration ';' { printf("Declaration\n"); }

| expression ';' { printf("value of expression: %d\n", $1); }

| VAR '=' expression ';' {

sym[$1] = $3;

printf("Value of the variable: %d\t\n",$3);

}

| SWITCH '(' VAR ')' START Bases END

| FOR '(' NUM THREEDOT NUM ')' START statement END {

int a = $3;

int b = $5;

int i;

for(i=a; i<=b; i++) {

printf("\nvalue of For Loop for %d time = %d\n",i, $8);

}

}

| IF '(' expression ')' START statement END %prec IFX {

if($3) {

printf("\nvalue of expression in IF: %d\n",$6);

}

else { printf("condition value zero in IF block\n");

}

}

| IF '(' expression ')' START statement END ELSE START statement END {

if($3) {

printf("value of expression in IF: %d\n",$6); }

else {

printf("value of expression in ELSE: %d\n",$10);

} }

;

Bases: Base

| Base Defaults

;

Base : /\*NULL\*/

| Base Cases

;

Cases : CASE NUM ':' expression ';' BREAK ';' { printf("Value of the case expression: %d\t\n",$4); }

;

Defaults : DEFAULT ':' expression ';' BREAK ';' { printf("Value of the default expression: %d\t\n",$3); }

;

declaration : TYPE ID1

;

TYPE : INT

| FLOAT

| CHAR

;

ID1 : ID1 ',' VAR

|VAR

;

expression: NUM { $$ = $1; }

| VAR { $$ = sym[$1]; }

| expression '+' expression { $$ = $1 + $3; }

| expression '-' expression { $$ = $1 - $3; }

| expression '\*' expression { $$ = $1 \* $3; }

| expression '/' expression { if($3) {

$$ = $1 / $3;

}

else {

$$ = 0;

printf("\ndivision by zero\t");

}

}

| '-' expression %prec UMINUS { $2 = - $2; }

| expression '^' expression { $$ = pow($1, $3); }

| expression '<' expression { $$ = $1 < $3; }

| expression '>' expression { $$ = $1 > $3; }

| '(' expression ')' { $$ = $2; }

| SIN '(' expression ')' { double val = sin($3 \* PI / 180); printf("sin %d = %lf\n",$3, val); $$ = val;}

| COS '(' expression ')' { double val = cos($3 \* PI / 180); printf("cos %d = %lf\n",$3, val); $$ = val; }

| TAN '(' expression ')' { double val = tan($3 \* PI / 180); printf("tan %d = %lf\n",$3, val); $$ = val; }

| LN '(' expression ')' { double val = log($3 \* PI / 180); printf("ln %lf\n",$3, val); $$ = val; }

;

%%

int yywrap() {

return 1;

}

yyerror(char \*s){

printf( "%s\n", s);

}

Output file

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Multi Line Comment

Single Line Comment

Single Line Comment

Declaration

Declaration

Single Line Comment

Value of the variable: 10

Value of the variable: 20

Value of the variable: 30

Single Line Comment

sin 60 = 0.866025

value of expression: 0

cos 60 = 0.500000

value of expression: 0

tan 60 = 1.732051

value of expression: 1

tan 60 = 1.732051

value of expression: 1

value of expression: 18

value of expression: -3

Value of the variable: 30

Value of the variable: 10

Single Line Comment

value of expression: 12

value of expression: 33

value of expression in ELSE: 33

value of expression: 26

value of expression in ELSE: 26

Value of the case expression: 30

Value of the case expression: 50

Value of the case expression: 40

Value of the default expression: 20

value of expression: 10

value of expression: 20

value of expression: 30

value of expression: 30

Value of the variable: 5

value of expression: 6

value of For Loop for 1 time = 6

value of For Loop for 2 time = 6

value of For Loop for 3 time = 6

value of For Loop for 4 time = 6

value of For Loop for 5 time = 6